Comparing Applications with the Software Problem Area Taxonomy

Maj Brandon Froberg MSgt Erik Olson Mr. Duane Baldish

***Abstract*— There are numerous methods and models to type, classify, describe, and compare software applications. Yet, there is a lack of software taxonomies, which categories software applications in terms of problems and solution spaces. We created a software problem taxonomy for use within Air Force Special Operations Command (AFSOC). The novel taxonomy enables comparing and contrasting software applications, but specifically focuses on the problems the software applications aim to solve. Our approach enables more fair comparisons of applications to solve or fulfill requirements and/or capability gaps.**

I. BACKGROUND

AFSOC, and the Department of Defense writ large, has capability gap to accurately compare and contrast available software solutions via the problems they solve. Situations arise in which software applications are directly compared by using descriptive attributes (e.g., a categorization of career/functional area, or development method). Often, comparisons of attributes are irrelevant, because they fail to consider the problem space first. For instance, "Enterprise Level" solutions are treated as a one size fits all application based only on descriptive attributes. Fair comparisons should first consider the problem space the software application is used to solve.

II. RESEARCHING EXISTING TAXONOMIES

In our initial research, we did not encounter software classifying methods/models which considered problem spaces. Existing software classifications methods/models use descriptive attributes to type, classify, describe application capabilities. For instance, the seminal Forward & Lethbridge taxonomy is robust, but we feel it focuses on software capabilities and/or descriptive attributes[1]. We argue the taxonomy classifies software by implementations of tools, techniques, or procedures; thus, we created a method to classify software by the problems they solve:

Algorithm 1 Classifying Software by Problems

1: **procedure** MAPPING SOFWARE TO PROBLEM SPACE
2: **A** ←*software application/capabilities*
3: **B** ←*problem space category/sub-category*
4: *An* **A** *is used to solve a problem in* **B**

Comparatively, Forward & Lethbridge's taxonomy follows a method which mapped software into a domain of **A** (i.e., software capabilities) versus **B** (i.e., a problem space).

For example: A **decision support system** (A.bus.3) or **inventory control** (D.im.1) app is used to solve problem in **Data Analysis** and **Data Visualization**. Other research or taxonomies for software categorization showed a similar lack of describing apps by their respective problem spaces[2][3][4][5][6][7].

III. CREATING A "16TH STANDARD"

Randall Monroe, the author of xkcd comics, presents a satire on how standards proliferate:

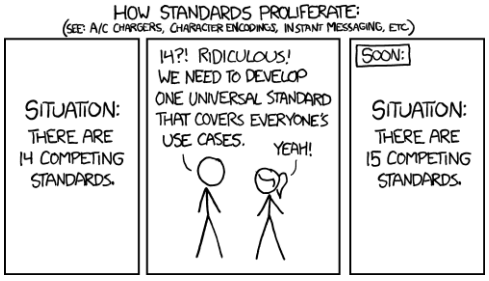

Fig. 1. XKCD's "Standards"[8]

Given the absence of software taxonomies, which focus on software problem spaces, we propose the Software Problem Area Taxonomy (SPAT):

Main Category	Sub-Category
Business Software	Presentation, Word Processing, Mathematical Spreadsheets
Computer Management	Command & Control (C2), Remote Access, Patching & Updates, Device Management
Data Analysis	Manual, Automatic, Machine Learning-based, AI-based
Data Collection	Primary, Secondary+, Archival
Digital Assembly	Drafting, Manufacturing, Multimedia, Visualization
Information Transfer	Audio, Video, Text, Raw
Information Visualization	Dashboarding, Summary
Modeling & Simulation	Manual, Automatic, Model Generation/Refinement, Visualization, Wargaming
Workflow & Processing	Manual, Automatic Non-repudiation Receipt

TABLE I
A SUBSET OF SPAT VERSION 0.1.0 CATEGORIES

We present version 0.1.0, using Semantic Versioning to describe Major, Minor, and Patch releases. Our proposition is not an all-encompassing list. Also, we fully expect the main and sub categories to expand, and undergo further refinement, as the larger Department of Defense and industry partners communities review this taxonomy.

Another important consideration to the taxonomy is to enable, capture, and describe software in terms of metadata attributes. Attributes are largely captured in existing taxonomies, inclusion further extends the qualification of application taxonomy mapping. Joining the problem space category to applicable attributes gives broader capabilities to sort, type, and

categorize apps. For instance Data Analysis and Information Visualization categories may contain Aircraft Logistical Supply and Computer Logistics Management apps, yet including metadata attributes permits a differentiatiion of the apps. Thus, the developed taxonomy includes collecting metadata attributes, and a subset is shown below:

Attribute	Example Descriptor
Development Technique	Traditional, Agile, DevSecOps Pipeline
Security	Manual, Automatic, Machine Learning-based, AI-based
Functional Area	Mission Planning, C2 Communications, Computers, Financial, Acquisition, Development, Intelligence, Surveillance, Reconnaissance
Code Level	Low Level/Binary, Bytecode, Intermediate representation, Script, No-code
Rendering	2D, 3D, Augmented

TABLE II
EXAMPLE SOFTWARE APPLICATION METADATA ATTRIBUTES

IV. USING THE TAXONOMY

Currently the taxonomy is used for immediate comparisons in Analysis of Alternatives and In-progress Reviews within the AFSOC Headquarters. We have several application comparisons that are in progress. The main comparison is of Robotic Process Automation (RPA) tools. Using our taxonomy shows how RPA tools are not one-to-one. Moreover, the taxonomy enables a clear and separate comparison of tools that can directly be used to justify funding and project continuation and/or support.

Additionally, AFSOC has an education & training initiative which enables and empowers citizen developers in the command. Our taxonomy is being used as an education tool to assist our trainees while they draft capability/business needs statements. This education facilitates the identification of requirements and essential capabilities prior to application development.

V. FUTURE WORK

The focus of this effort is intended to support software applications comparisons. The proposed taxonomy allows individuals to sort and categorize an application into an appropriate "problem-solution-spaces" which simplifies and clarifies apples-to-apples comparisons. We intend to leverage AFSOC corporate groups and subject matter experts to further refine our taxonomy within AFSOC.

Ultimately, a Neo4j, graph database reference, will be built, for the categories and attributes, which will be shared on public resources (e.g., GitHub.com) and will signify the official release of SPAT version 1.0.0.

REFERENCES

[1] Andrew Forward and Timothy C Lethbridge. "A taxonomy of software types to facilitate search and evidence-based software engineering". In: *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*. 2008, pp. 179–191.

[2] Andrea Capiluppi and Nemitari Ajienka. "Towards A Dependency-Driven Taxonomy of Software Types". In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 2020, pp. 687–694.

[3] Jose Calvo-Manzano Villalón et al. "A taxonomy for software testing projects". In: *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE. 2015, pp. 1–6.

[4] Jeffrey M Barnes. "Software architecture evolution". PhD thesis. Carnegie Mellon University, 2013.

[5] Mark Kasunic. *A data specification for software project performance measures: Results of a collaboration on performance measurement*. Tech. rep. Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2008.

[6] Muhammad Usman. "Supporting effort estimation in agile software development". PhD thesis. Blekinge Tekniska Högskola, 2015.

[7] Viktoria Stray et al. *Agile Processes in Software Engineering and Extreme Programming: 21st International Conference on Agile Software Development, XP 2020, Copenhagen, Denmark, June 8–12, 2020, Proceedings*. Springer Nature, 2020.

[8] Randall Munroe. *Standards*. URL: `https://xkcd.com/927/`.